- **ISS 3102 EXPERT SYSTEMS**
  - **TOPIC 1 – Introduction to Expert Systems**
  - **TOPIC 2 – Domains of Application**
  - **TOPIC 3 – Expert System Architecture**
  - **TOPIC 4 – Knowledge Engineering**
  - ***TOPIC 5 - Uncertainty in Expert Systems***
  - **TOPIC 6 – Building Expert Systems**
  - **TOPIC 7 – Limitations & Problems**
  - **Topic 8 – Recent Work & Trends**
  - ***Final Class Presentations***

LABWORK

1. Running a Knowledge-Based System (KBS)
2. Knowledge Acquisition & Inference Demonstration
3. User Interface Development
4. Project Presentations

- **LECTURER : DOCTOR GONGERA,** *MSc.; BSc.; AI(MIT), CISM ;GDPR,MCSE; CCNA™; Net Security; PhD Student.*
- **HOURS: FRIDAY**
- **MODE: VIRTUAL /ZOOM**

Introduction

- Expert Systems are a branch of Artificial Intelligence (AI) concerned with developing computer programs that emulate the decision-making ability of human experts in specialized domains such as medicine, agriculture, finance, and engineering
- They form a core part of **Knowledge-Based Systems (KBS)** — systems that use structured knowledge representations rather than just algorithmic code.

❑ **Motivation for Expert Systems -** Expert Systems emerged due to:

- ✓ scarcity of domain experts,
- ✓ need for consistent decisions,
- ✓ high cost of expert labor,
- ✓ preservation of rare knowledge,
- ✓ demand for automation in complex decision tasks.

Application in Kenya -

- healthcare (county hospitals),
- agriculture (extension officers),
- veterinary services,
- financial advisory,
- legal and compliance domains.

✓ Expert systems can provide 24/7 decision support and standardized expertise.

Characteristics of Expert Systems

- High Performance — approaches expert-level decisions.
- Domain Specificity — narrow, specialized problem domains.
- Rule-Based Reasoning — uses structured production rules.
- Explanatory Ability — system can justify conclusions.
- Uncertainty Handling — supports reasoning under incomplete or uncertain data.
- User-Friendly Interface — supports non-technical users.
- Consistency — decisions do not vary with mood, fatigue, or bias.

Suitable Domains for Expert Systems

Agriculture & Veterinary
- ✓ Crop nutrient deficiency diagnosis
- ✓ Pest and disease identification
- ✓ Livestock disease advisor

Medical Diagnosis
- ✓ Symptom-based diagnosis
- ✓ Treatment recommendation
- ✓ Triage support

- specialized expert knowledge,
- well-defined rule structures,
- limited ambiguity
- clear problem constraints.

Industrial & Engineering
- ✓ Machine fault diagnosis
- ✓ Process control
- ✓ Equipment maintenance prediction

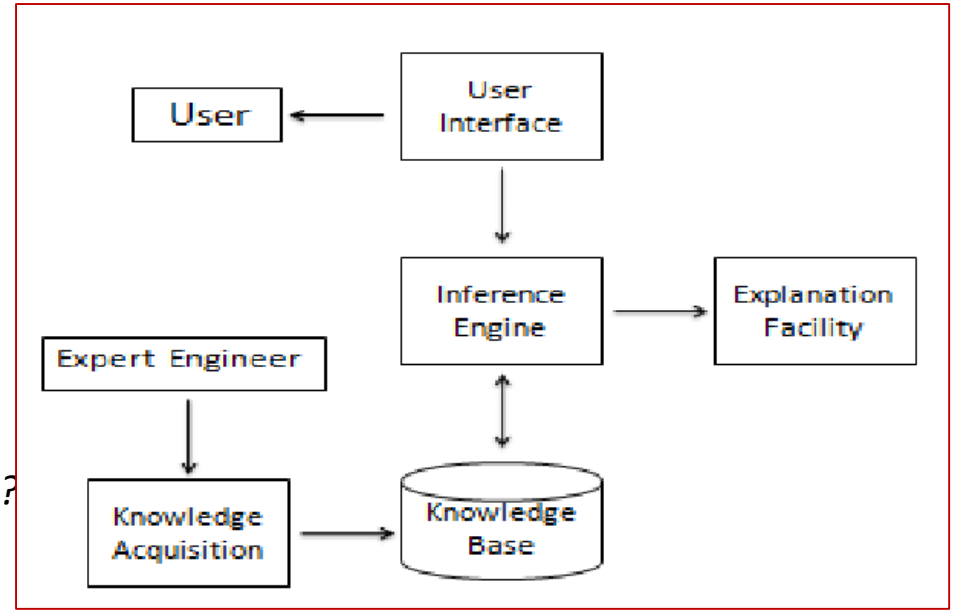
Financial Decision Support
- ✓ Credit scoring
- ✓ Loan portfolio screening
- ✓ Fraud detection

Expert Systems vs Conventional Systems

Feature	Conventional Systems	Expert Systems
Knowledge Source	Programmer	Domain Expert
Logic	Algorithmic	Knowledge-based
Reasoning	Procedural	Inference-based
Domain	General	Narrow
Adaptability	Low	Moderate (with re-knowledge)
Explanation	None	Explanation facility
Learning	None	Optional (with hybrid models)

Components of an Expert System

1. **Knowledge Base**
    1. domain facts + heuristic rules
2. **Inference Engine**
    1. applies rules to draw conclusions
    2. supports:
        1. forward chaining
        2. backward chaining
3. **Working Memory**
    1. temporary facts during reasoning
4. **User Interface**
    1. handles interaction with users
5. **Explanation Facility**
    1. answers:
        1. *"Why was this conclusion reached?*
        2. *"How was it derived?"*
6. **Knowledge Acquisition Module**
    1. tools for capturing expert knowledge
7. **Expert System Shell**
    1. framework without domain knowledge
    2. allows domain-specific rule filling

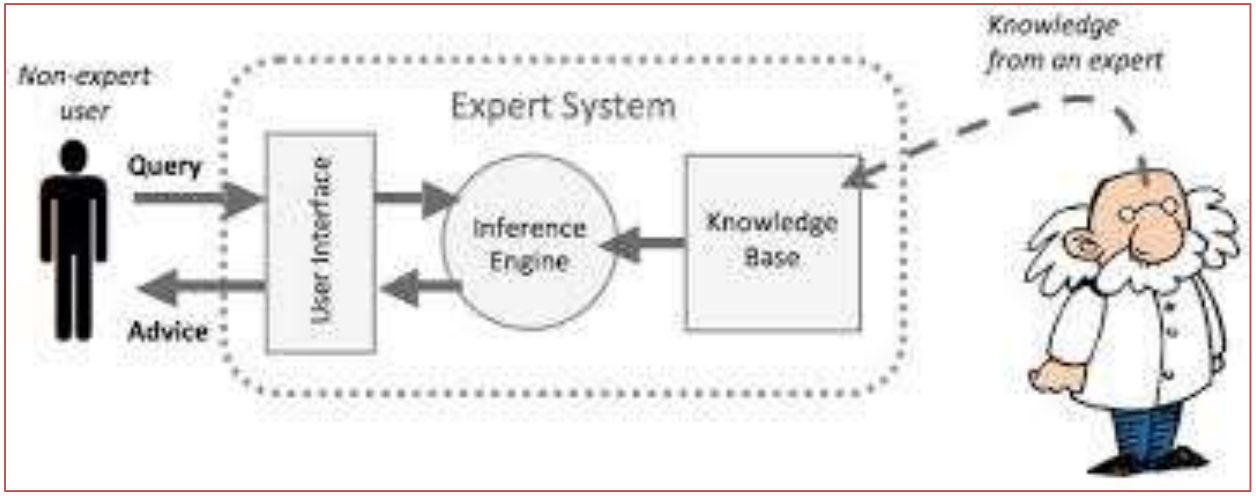


Expert System Shells provide:

- inference engines,
- UI components,
- rule management tools.

How Expert Systems Work

1. User provides facts (input).
2. Working memory stores current knowledge.
3. Inference engine matches facts with rules.
4. Conclusions or recommendations are derived.
5. Explanation facility clarifies reasoning.
6. User receives final output or advice.



Limitations & Problems of Expert Systems

1. Knowledge Acquisition Bottleneck - extracting tacit knowledge from experts is slow and difficult.
2. Maintenance Complexity - updating rules can break consistency.
3. Limited Scope -cannot solve problems outside defined domains.
4. Lack of Intuition - unlike human experts, cannot improvise.
5. Performance Overheads - scaling rule bases increases inference cost.
6. Uncertainty Handling - not all expert systems handle uncertainty well.
7. User Trust & Acceptance -especially in sensitive domains (medicine, law).

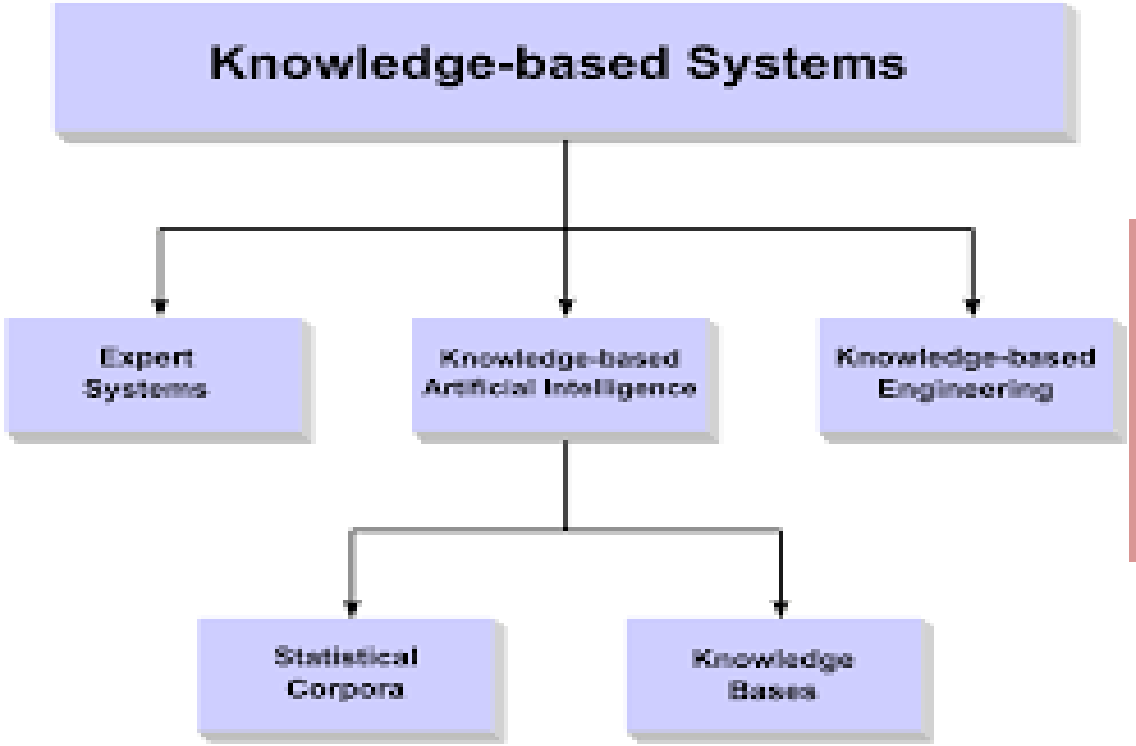
Recent Work in Expert Systems

- ✓ machine learning,
- ✓ data mining,
- ✓ fuzzy logic,
- ✓ Bayesian methods,
- ✓ natural language interfaces,
- ✓ neuro-symbolic AI.

Trends include:

- hybrid intelligent systems,
- explainable AI (XAI),
- domain-specific reasoning (e.g., fintech compliance),
- medical expert systems with uncertainty scoring.

Relationship to Knowledge-Based Systems(KBS)



KBS include:

- ✓ Expert Systems
- ✓ Recommender Systems
- ✓ Case-Based Reasoning Systems
- ✓ Knowledge Management Systems

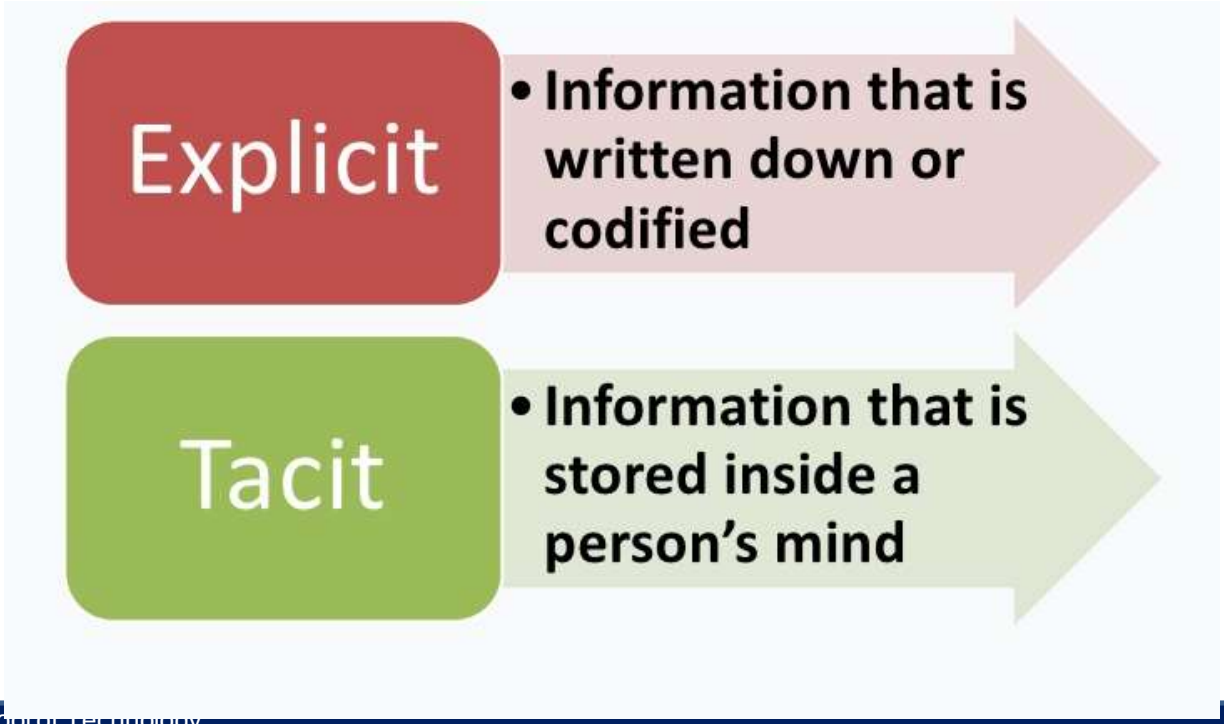
❑ Expert Systems are a subset of Knowledge-Based Systems (KBS).

Quiz

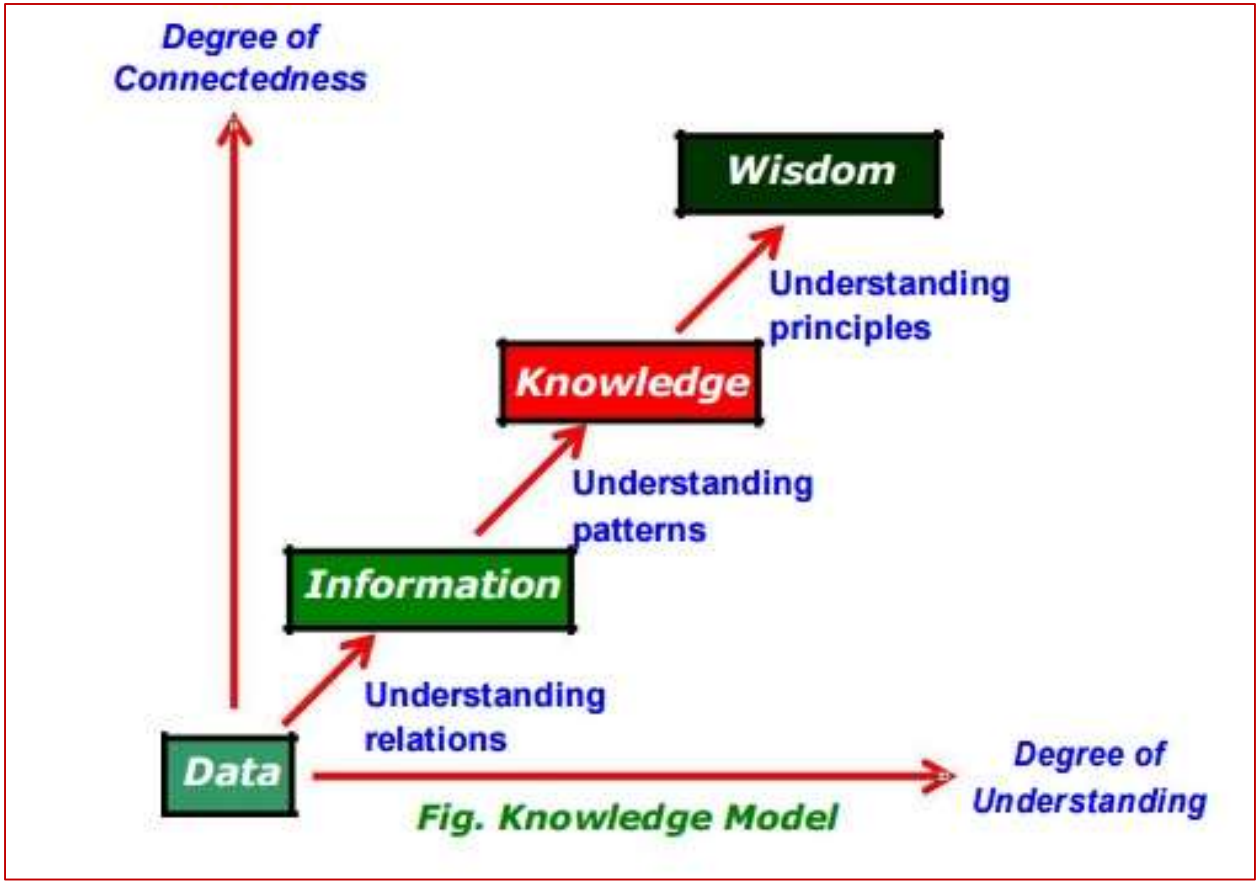
1. Define an Expert System and explain its purpose.
2. Identify three Kenyan domains suitable for Expert Systems and justify them
3. Describe four components of an Expert System.
4. Explain the knowledge acquisition bottleneck.
5. Compare Expert Systems with conventional algorithmic systems.

Types of Knowledge

- Knowledge Representation (KR) refers to the methods used to formally encode expert domain knowledge so that a machine can reason with it. Expert Systems rely on structured domain knowledge supplied by experts.
- KR is central because the quality of representation directly impacts system performance, correctness, and usability.



Knowledge Model



- ✓ A knowledge model tells, that as the degree of "connectedness" and "understanding" increases, we progress from data through information and knowledge to wisdom.

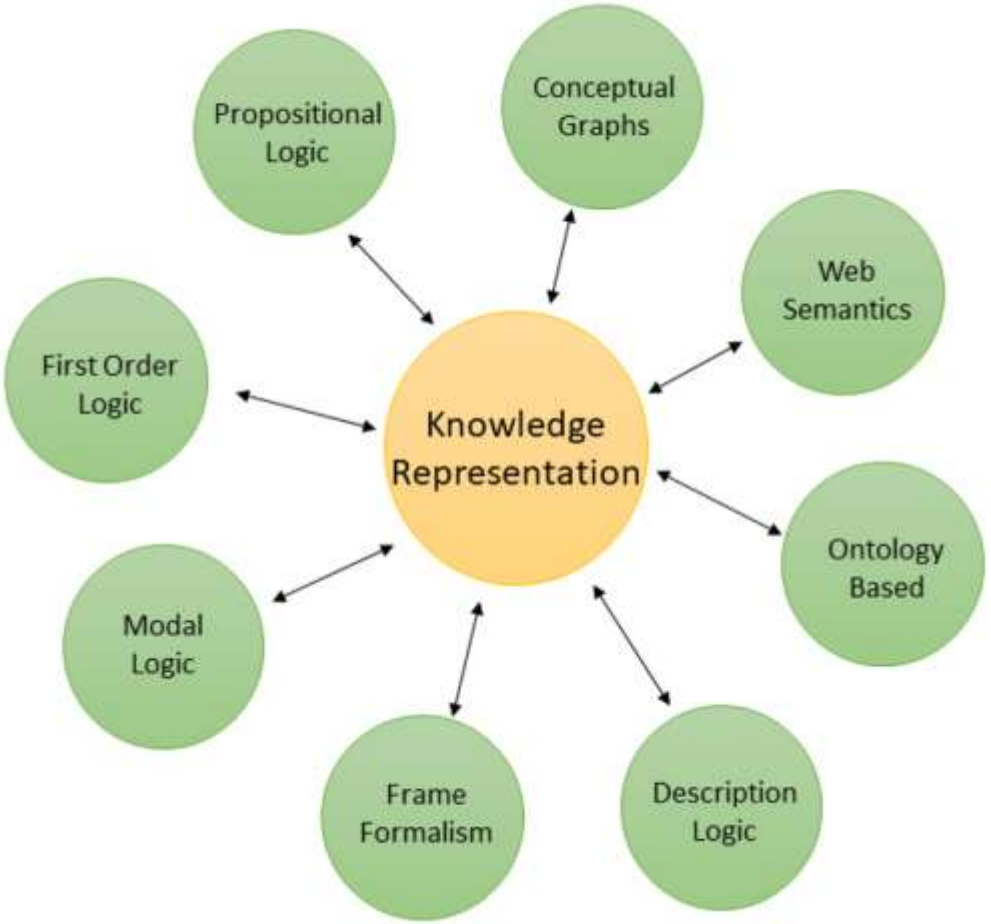
Types of Knowledge

Expert Systems encode several classes of knowledge:

1. Declarative Knowledge - Facts about the world. Example: “Cows produce milk.”
2. Procedural Knowledge - How-to knowledge (methods, processes). Example: “To diagnose mastitis, perform a CMT test.”
3. Heuristic Knowledge - Rules of thumb used by experts. Example: “If leaves are yellow from the tip, suspect nitrogen deficiency.”
4. Meta-Knowledge - Knowledge about controlling other knowledge. Example: “Check soil moisture before checking nutrient deficiency rules.”
5. Structural Knowledge - Relationships among concepts. Example: “Foot-and-Mouth Disease affects cloven-hoofed animals.”

Knowledge Representation Schemes

1. Logic-Based Representation
2. Production Rules
3. Frames
4. Semantic Networks
5. Decision Tables
6. Ontologies
7. Fuzzy Knowledge
8. Bayesian Knowledge
9. Hybrid Representations



Logic-Based Representation

Based on propositional or predicate logic. Used mostly for:

- legal reasoning,
- auditing compliance,
- theorem proving.

Example (predicate logic): Rain(Nakuru) → Wet Soil(Nakuru)

Strengths:

- rigorous, formal, unambiguous
- proof and verification supported

Weaknesses:

- poor for heuristic domains
- limited handling of uncertainty

Production Rules (IF–THEN Rules)

Most common representation in classical Expert Systems.

General form:
IF <condition> THEN <action/conclusion>
Example (Agriculture):
IF maize leaves are yellow AND soil moisture is low THEN diagnose nitrogen deficiency.

Advantages:

- intuitive for experts
- modular (rules added easily)
- supports chaining

Disadvantages:

- rule explosion at scale
- maintenance complexity increases

Frames

Object-like structures that represent stereotypical knowledge.

Example Frame:

```
Frame: Cow
  Slots:
    Species = Bos taurus
    Diet = Grass
    Diseases = {Mastitis, FMD}
```

Supports inheritance:
Dairy Cow → Cow

Useful for:

- veterinary systems
- agricultural modeling
- manufacturing diagnostics

Semantic Networks

Graph-based concept relations.

Nodes = objects or concepts
Edges = relationships

Example:

```
Coffee → is A → Crop
Coffee → grows In → high Altitude
```

Decision Tables

Tabular form for rule sets with multiple conditions.
Useful in:

- finance loan approvals,
- compliance,
- SACCO rules.

Ontologies

Ontologies define structured vocabularies and relationships within a domain.

Recent Kenyan/African ontology work includes:

- agricultural plant disease ontologies,
- livestock health ontologies,
- Swahili linguistic ontologies.

Useful for:

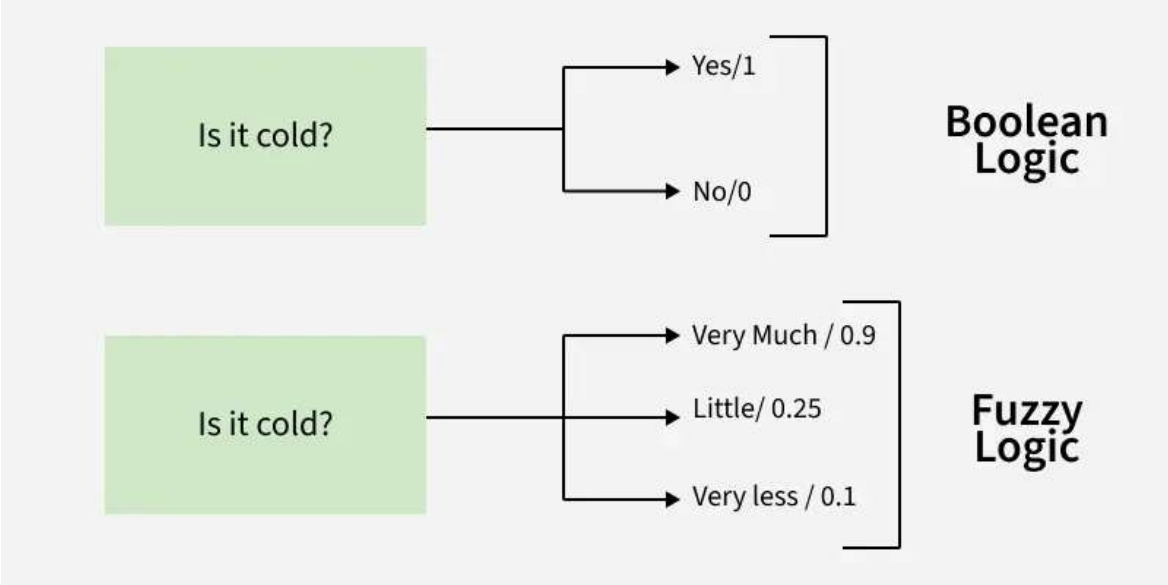
- interoperability,
- semantic web,
- knowledge sharing.

Fuzzy Knowledge Representation

Used for vague or linguistic concepts such as:

- "low soil moisture,"
- "high temperature,"
- "moderate acidity."

Supports decision-making where boundaries are not crisp.



Bayesian Knowledge Representation

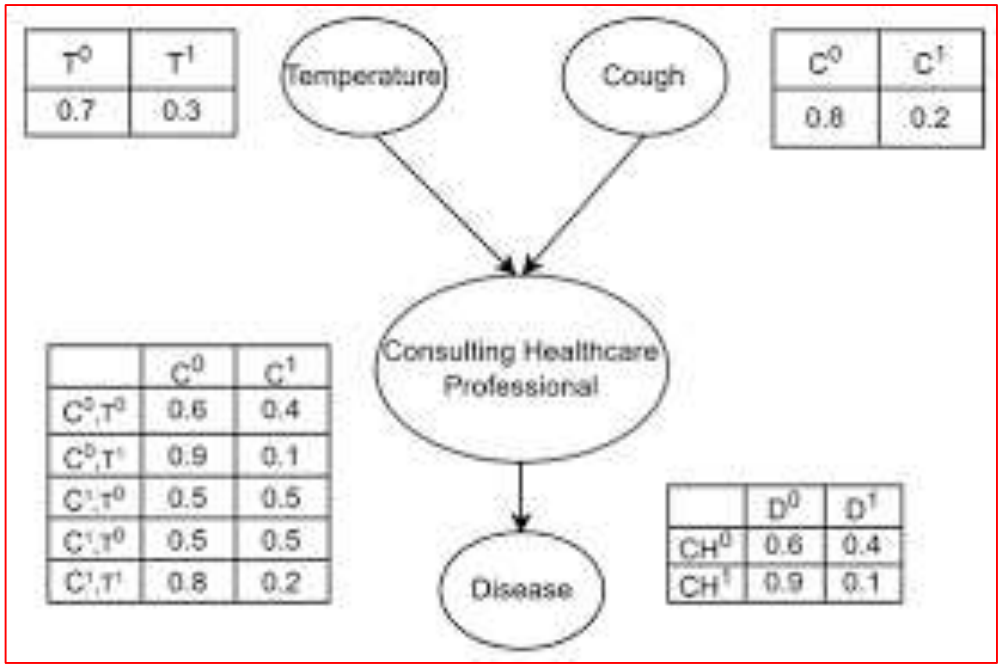
Probabilistic representation addressing uncertainty.

Example:

```
P(Malaria | Fever, Chills)
```

Used in:

- clinical decision support,
- risk assessment,
- financial scoring.



Hybrid Representations

Combine two or more methods.

Common hybrids include:

- rule + frame,
- rule + fuzzy,
- neural + symbolic.

Modern systems often blend ML with symbolic reasoning.

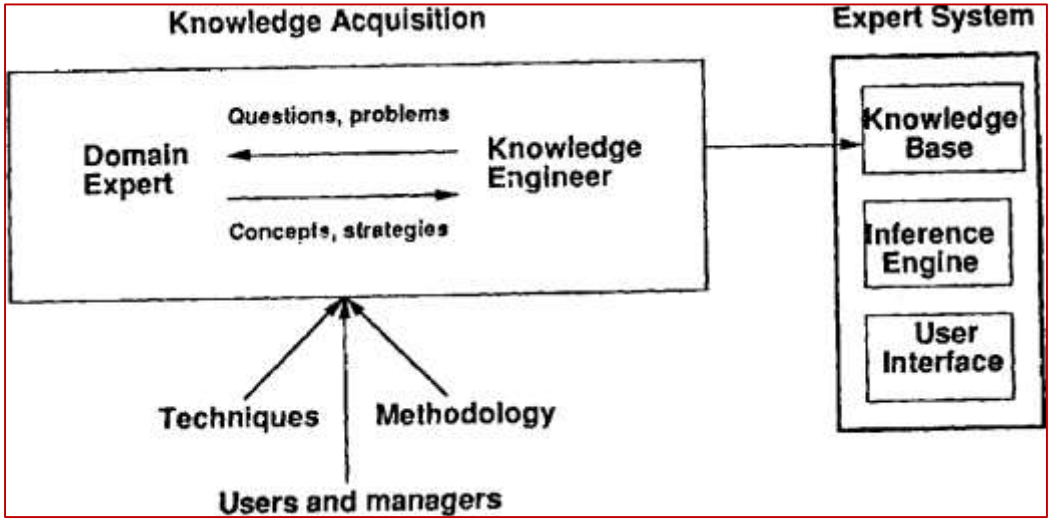
Knowledge Acquisition

Knowledge acquisition deals with obtaining knowledge from sources such as:

- •domain experts,
- •textbooks,
- •scientific studies,
- •data mining,
- •machine learning models,
- •field observation.

Challenges ("Knowledge Acquisition Bottleneck"):

- •experts are busy,
- •knowledge may be tacit - Tacit Knowledge: Knowledge that is difficult to transfer to another person by means of writing it down or verbalizing it.
- •human experts disagree,
- •knowledge expressed informally,
- •translation to computational form non-trivial.



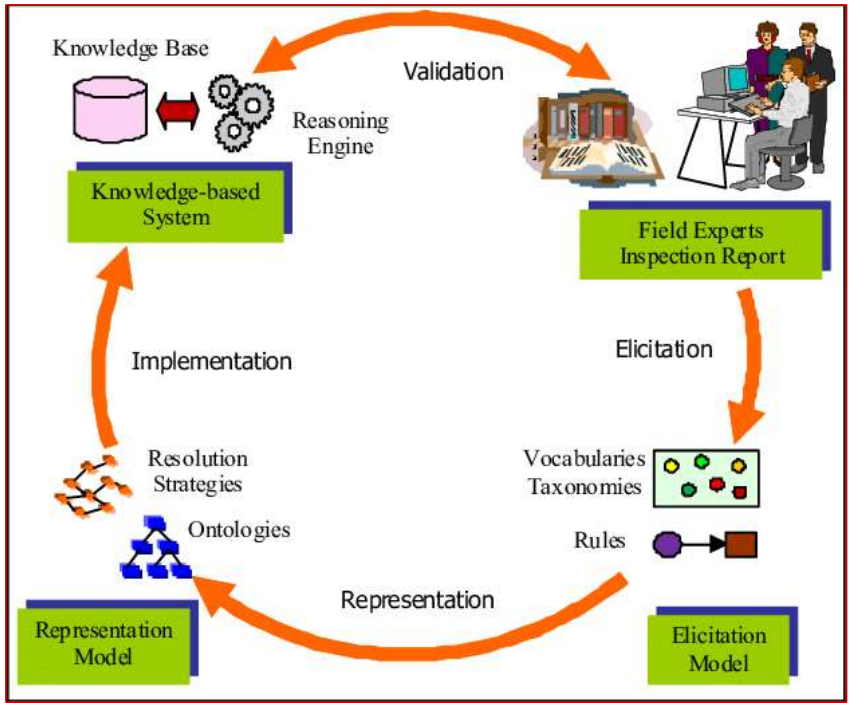
Knowledge Validation & Verification

After acquisition, the knowledge must be checked:

- **Verification** — Are we building the system right? (internal correctness)
- **Validation** — Are we building the right system? (matches real-world expectations)

Validation methods include:

- field testing,
- expert review,
- simulation,
- test cases,
- comparison with human expert decisions.



Quiz

1. **D**efine knowledge representation and explain its importance.
2. Differentiate declarative and procedural knowledge with examples.
3. Discuss advantages and disadvantages of rule-based representation.
4. Explain the knowledge acquisition bottleneck.
5. Using examples, compare semantic networks and frames.

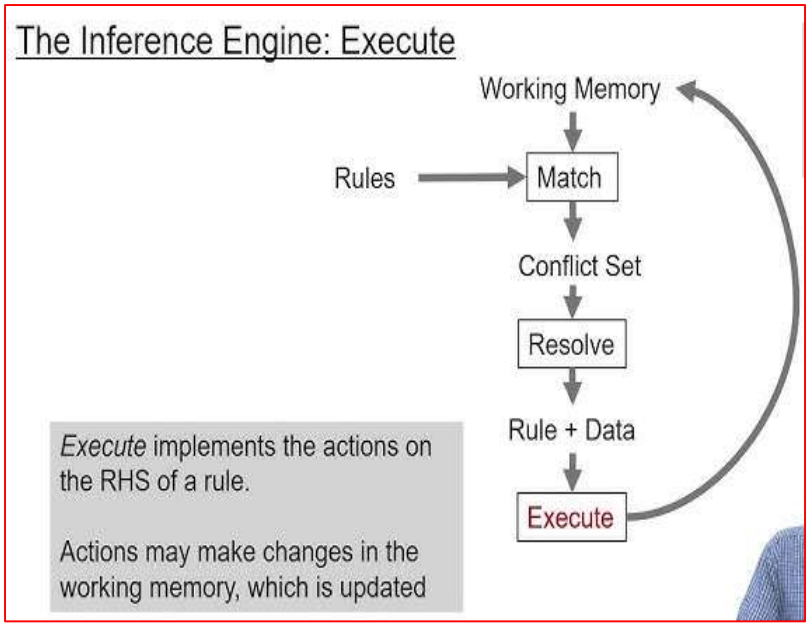
INFERENCE ENGINE

- Inference is the mechanism by which an Expert System applies knowledge (rules, facts, relationships) to reach conclusions, provide recommendations, or diagnose a problem.
- Inference strategies determine **how** and **in what order** rules are examined, matched, and fired during problem-solving
- **The Inference Engine performs the following tasks**:

1. Match facts with rules
2. Select applicable rules
3. Resolve conflicts when multiple rules apply
4. Execute rule actions
5. Update working memory
6. Explain reasoning steps (optional)

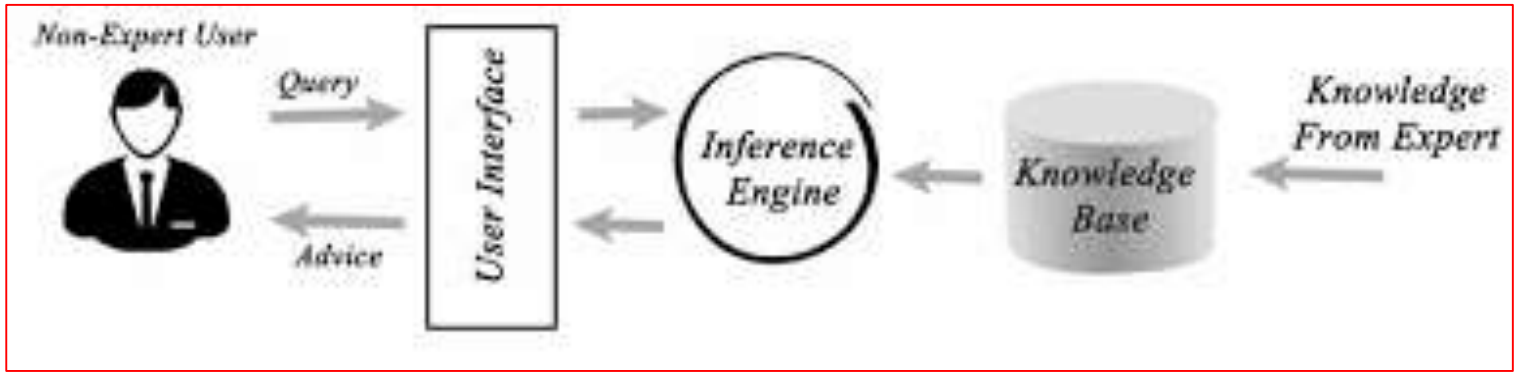
The process repeats until:

- a goal is achieved,
- no more rules apply,
- or the system halts.



Inference Strategies

- Inference strategies determine how and in what order rules are examined, matched, and fired during problem-solving
- **There are two primary inference strategies**:
  1. Forward Chaining (Data-Driven)
  2. Backward Chaining (Goal-Driven)
  3. Hybrid Strategies (combining both)



✓ Forward chaining suits **diagnosis**, while backward chaining suits **goal verification**.

Forward Chaining (Data-Driven)

- Forward chaining is an inference strategy that starts from known facts and applies rules to infer new facts until a conclusion is reached.

Operational Flow

1.Collect initial facts.

2.Match facts to rule antecedents.

3.Fire rules whose conditions are satisfied.

4.Add new facts to working memory.

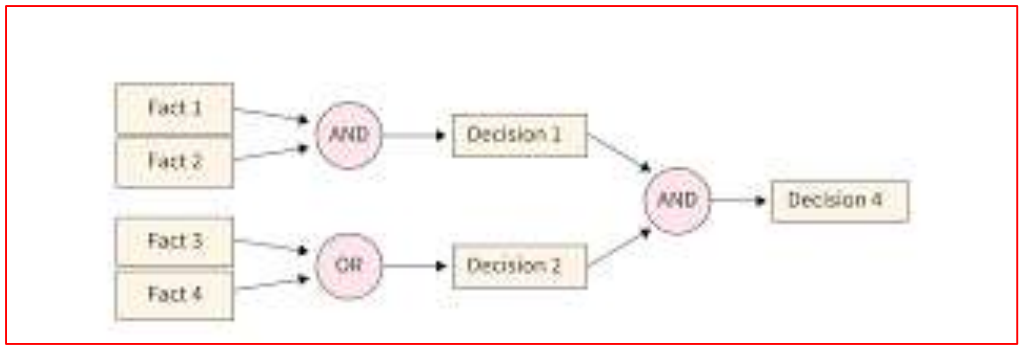
5.Repeat until no more rules apply or target reached.

Example (Agriculture)

Facts:
Soil Moisture = low
leaves = yellow

Rule:
IF soil Moisture = low AND
leaves = yellow
THEN deficiency = nitrogen

Result:
deficiency = nitrogen



Forward Chaining (Data-Driven)

Characteristics

- **Data-driven**
- Useful when input data is available upfront
- Explores multiple possible conclusions
- Suitable for diagnostic and monitoring tasks

Applications

✓ medical diagnosis systems

✓ crop/soil advisory systems

✓ equipment troubleshooting

✓ financial fraud detection

Advantages

- Handles wide search spaces
- Natural for diagnostic tasks
- Easy to implement in rule engines

Disadvantages

- May generate irrelevant conclusions
- Search can become broad without goal constraints
- Potentially computationally expensive

©

Backward Chaining (Goal-Driven)

- Backward chaining is an inference strategy that starts with a specific goal/hypothesis and works backward to determine if facts support it.

Operational Flow

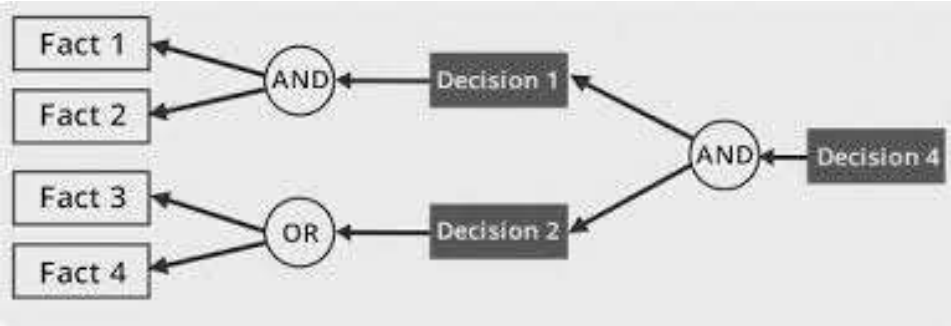
1.Set a goal (target conclusion).
2.Find rules that could satisfy the goal.
3.Check rule antecedents.
4.If antecedents unknown → treat them as subgoals.
5.Continue until:
   1. facts support the goal (success), or
   2. goal cannot be proven (failure).

Example (Finance)

Goal:
Approve Loan (customer A)

Rule:
IF income ≥ threshold AND collateral = adequate
THEN approve Loan(customer A)

Subgoals form:
•check income(customer A)
•check collateral(customer A)



Backward Chaining (Goal-Driven)

Characteristics

- **Goal-driven**
- Efficient when goals are narrow
- Minimizes unnecessary inference

Applications

✓ legal reasoning

✓ auditing & compliance systems

✓ credit approval systems

✓ medical treatment planning

Advantages

- Efficient for targeted reasoning
- Focused search
- Mirrors how experts justify decisions

Disadvantages

- Requires goals defined upfront
- Not ideal for discovery-type problems
- Can fail if key facts are missing

Hybrid Inference Strategies

- Hybrid strategies combine both data-driven and goal-driven approaches.

Common hybrid patterns:

•Forward generate → backward verify

•Backward trigger → forward propagate

•Cyclic feedback reasoning

Used in complex domains such as:

✓ clinical decision support

✓ industrial process control

✓ real-time planning

Conflict Resolution Strategies

When multiple rules are eligible to fire, the inference engine must choose which one to execute first. Common strategies include:

(a) Specificity Priority

•more specific rules fire before general ones

(b) Recency Priority

•rules using recently added facts fire first

(c) Rule Ordering

•rules arranged by domain expert priority

(d) Refraction

•prevents re-firing rules under identical conditions

(e) Salience/Weighting

•rules assigned priority scores

Conflict resolution is essential for:

- •system determinism,
- •efficiency,
- •traceability.

Termination Conditions

Inference ends when:

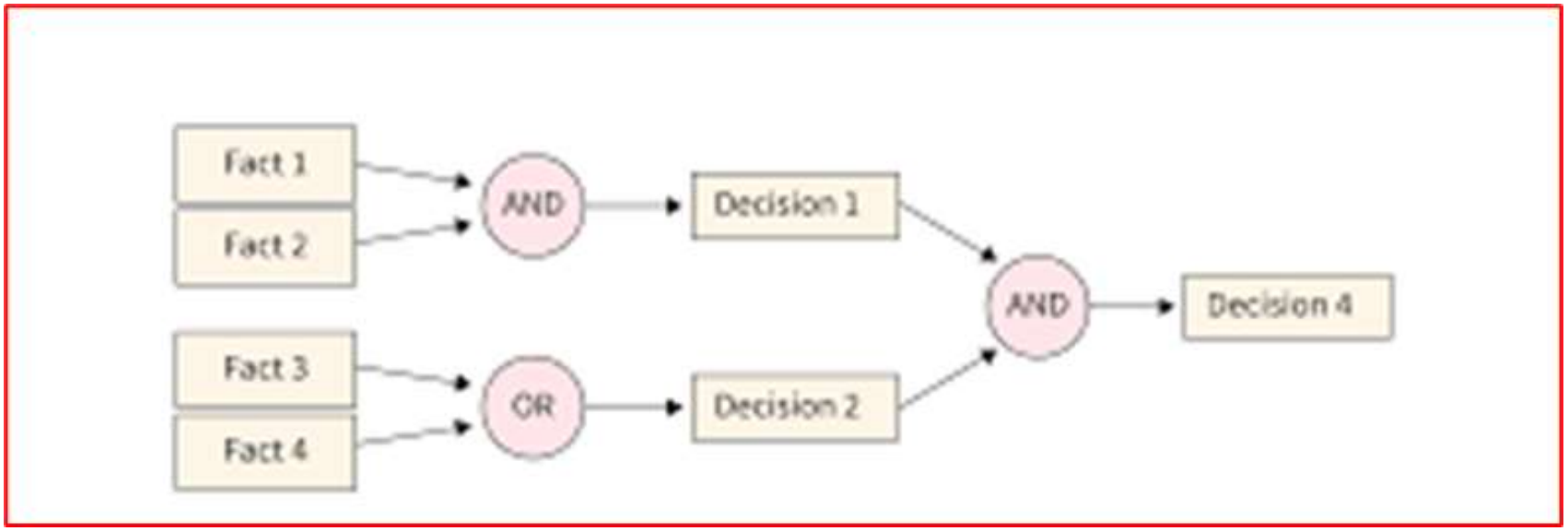
1.Goal achieved
2.No rule can fire
3.User halts process
4.Resource limits exceeded

Explanation Facility

Many Expert Systems provide "Why?" and "How?" explanations such as:

- **Why:** "Why did you ask for soil moisture?"
- **How:** "How did you diagnose nitrogen deficiency?"

This improves: transparency,user trust,adoption in sensitive fields.



Supplementary structures – Decision Tables & Trees in Inference

Decision Tables

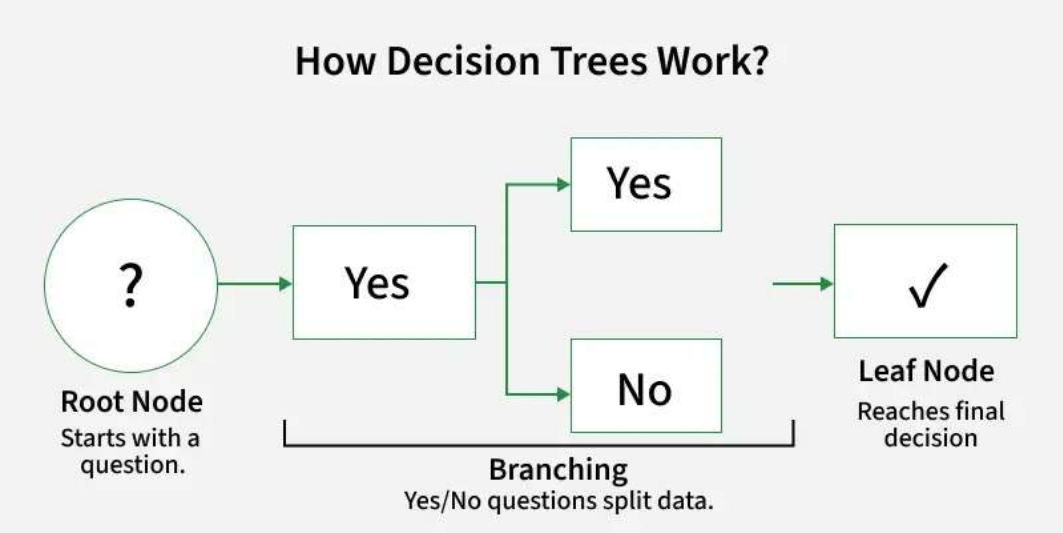
- Tabular format for rule combinations
- Useful in finance and compliance

Requirement Number					
Condition	1	2	3	4	5
User is authorized	F	T	T	T	T
Chemical is available	—	F	T	T	T
Chemical is hazardous	—	—	F	T	T
Requester is trained	—	—	—	F	T
Action					
Accept request			X		X
Reject request	X	X		X	

Decision Trees

- Graph format showing branching decisions
- Useful for diagnosis and classification

Both structures improve maintainability in systems where rules are numerous and combinatorial.



QUIZ

1. Compare forward and backward chaining.

2. Describe conflict resolution and list three resolution strategies.

3. Explain how backward chaining achieves goal-directed reasoning.

4. Give two Kenyan domains where forward chaining is useful and explain why.

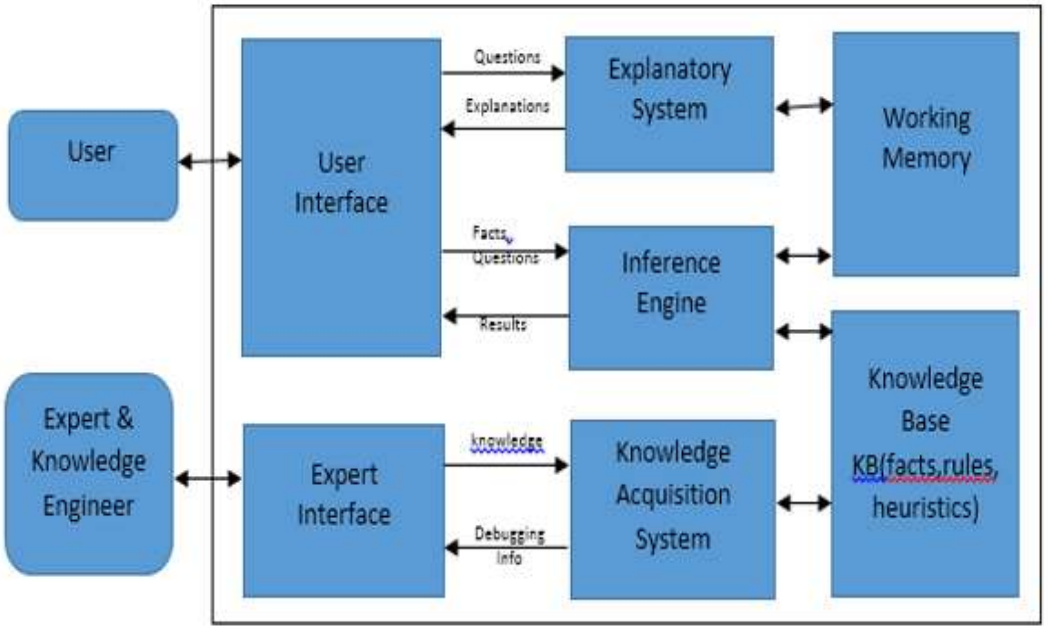
5. Discuss advantages of hybrid inference.

Architecture of an Expert System

A generic Expert System consists of the following core components:

1. Knowledge Base
2. Inference Engine
3. Working Memory
4. User Interface
5. Explanation Facility
6. Knowledge Acquisition Module
7. Expert System Shell

These are connected via an internal control cycle controlled by the inference engine.



ES Components - Review

1. **Knowledge base -** stores the domain-specific knowledge supplied by experts.
Contains: **facts** (declarative knowledge), **rules** (procedural/heuristic knowledge)
Knowledge types include: domain theories, heuristics, frames/ ontologies, decision tables.

2. Inference Engine - applies reasoning methods to derive conclusions from knowledge and facts. It performs: pattern matching, rule firing, conflict resolution, goal satisfaction.
Supports strategies: **forward chaining, backward chaining, hybrid inference**
The inference engine is the "brain" of the system.

3. **Working memory** stores: current facts provided by user, intermediate results, conclusions drawn during inference. Facts in working memory may change as new conclusions are inferred.

4. The User Interface (UI) enables information exchange between the system and user.
Two main functions: Input acquisition: questions, forms, data fields
Output display: conclusions, recommendations, explanations

ES Components - Review

5. Explanation Facility - Allows the system to answer: **Why** a question was asked, **How** a conclusion was reached
Example: *"I diagnosed maize streak because you reported mosaic leaves and stunted growth."*
This improves: trust, validation, compliance, student training, adoption in sensitive domains.

6. Knowledge Acquisition Module - This module assists in transferring knowledge from human experts into the knowledge base. Sources include: interviews, observation, domain manuals, machine learning extraction.
The difficulty of this task leads to the **knowledge acquisition bottleneck**.

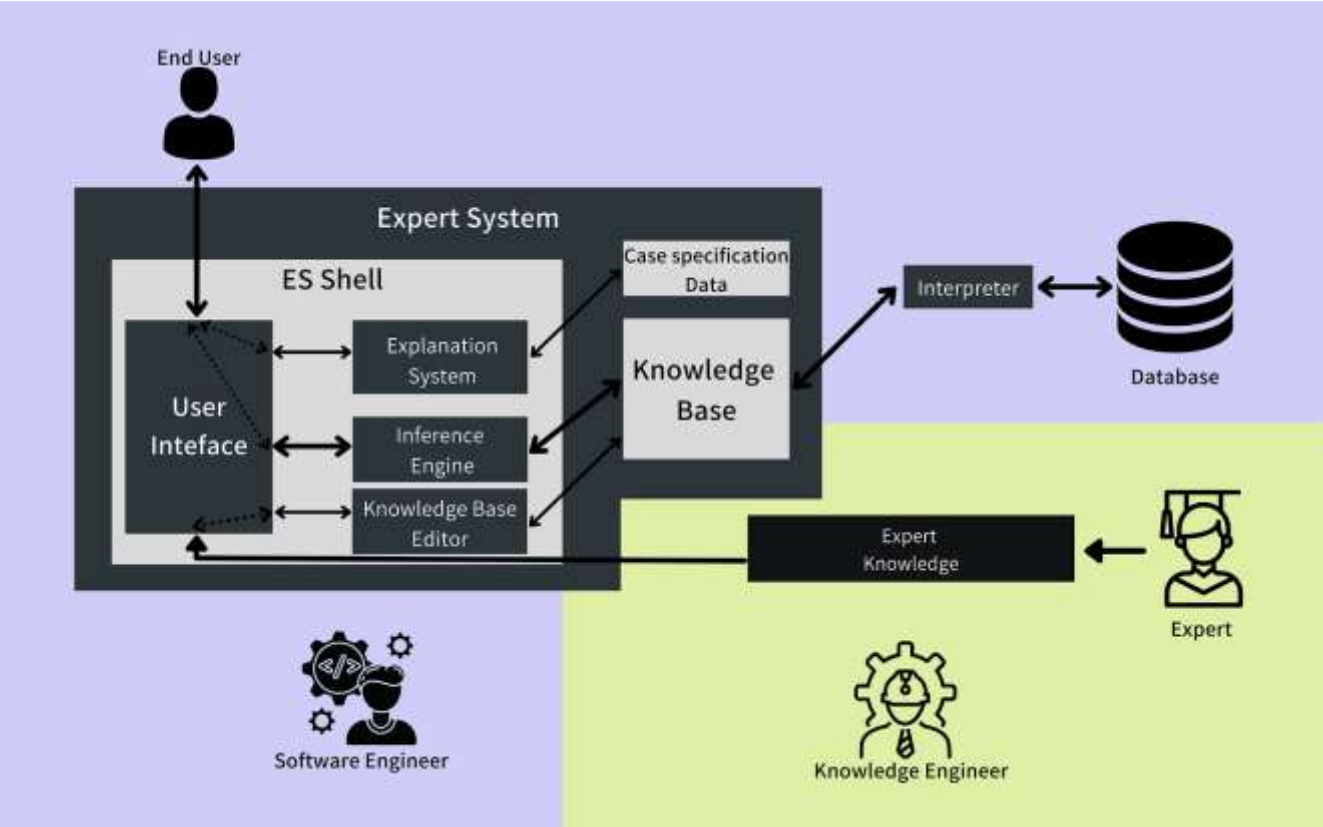
7. Expert System Shell - A Shell is an Expert System with no domain knowledge.
Contains: inference engine, explanation module, UI tools, rule editors.
The developer only inserts the domain rules.
Examples of shells: CLIPS, JESS, Prolog-based shells, Commercial medical shells, In-house enterprise shells

- CLIPS (C Language Integrated Production System)
- JESS (Java Expert System Shell)

✓ **Expert system shells are toolkits that can be used to develop expert systems. They consist of some built expert system components with an empty knowledge base. Hence, in most cases, the knowledge engineer is left with only populating the knowledge base**

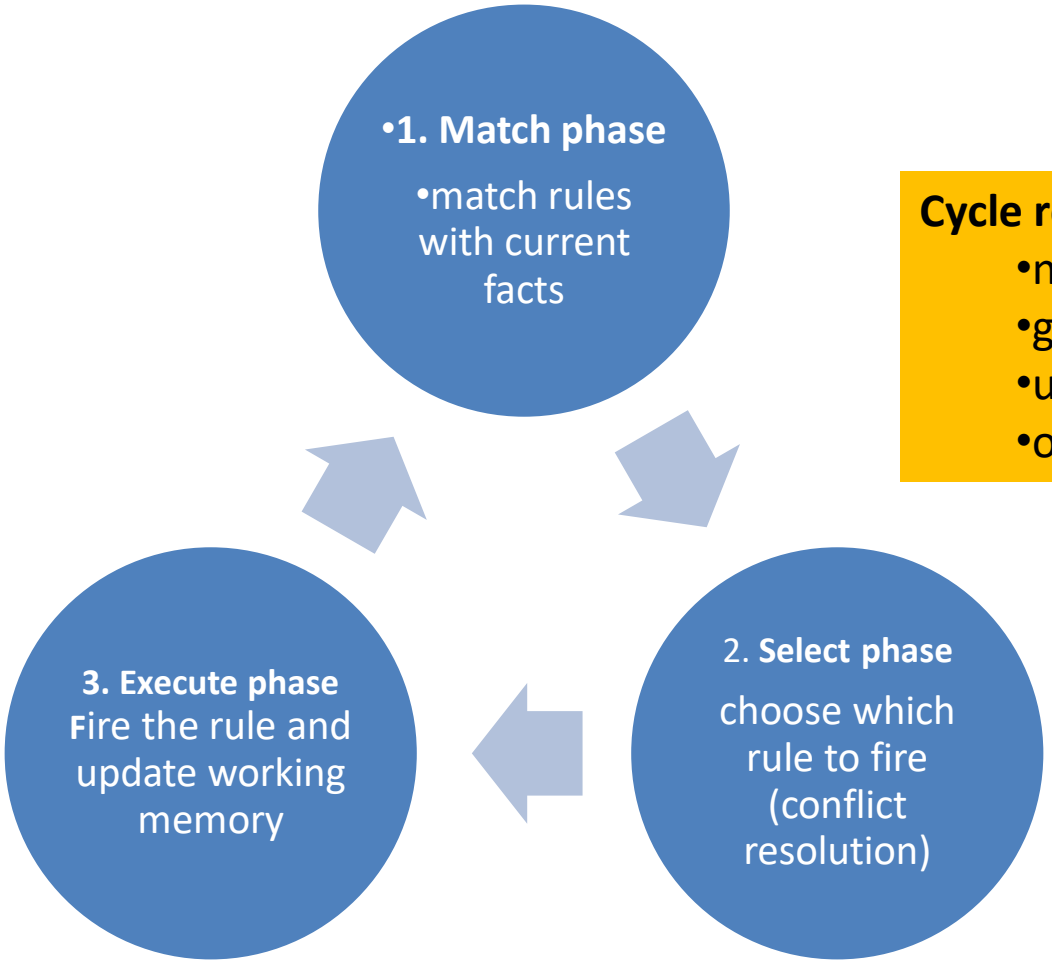
✓ Knowledge is populated using rules, frames or decision trees. The rules can be programmed by Prolog, Java, Python or any other preferred language

Expert System Shell



- ✓ ES Shells is a far better option rather than developing an expert system from the scratch. There are many ES Shells that support different programming languages that have been developed which open gates for developers who are familiar with various programming languages. The ES Shells should be properly selected mainly considering the requirements and the scale/size/complexity of the expert system to be developed.

Control Cycle in Expert Systems


•1. Match phase
•match rules with current facts
2. Select phase
choose which rule to fire (conflict resolution)
3. Execute phase
Fire the rule and update working memory

Cycle repeats until:

- •no rules match,
- •goal achieved,
- •user stops,
- •or resources exhausted.

Knowledge Organization

- rule bases,
- frame hierarchies,
- semantic networks,
- ontologies,
- Indexed fact databases
- etc.

Organization affects:

•search performance,

•inference efficiency,

•updating & maintenance.

Interaction with External Systems

Modern Expert Systems may integrate with:

- databases (SQL, NoSQL),
- sensors (in agricultural IoT contexts),
- ML models (hybrid systems),
- enterprise systems (finance, audits),
- cloud APIs.

Examples in Kenyan context:

✓ SACCO loan screening using rules + credit data

✓ crop advisory systems using satellite/soil sensors

✓ hospital triage systems using medical databases

Expert System Development Life Cycle

Design Considerations:

1.Domain complexity

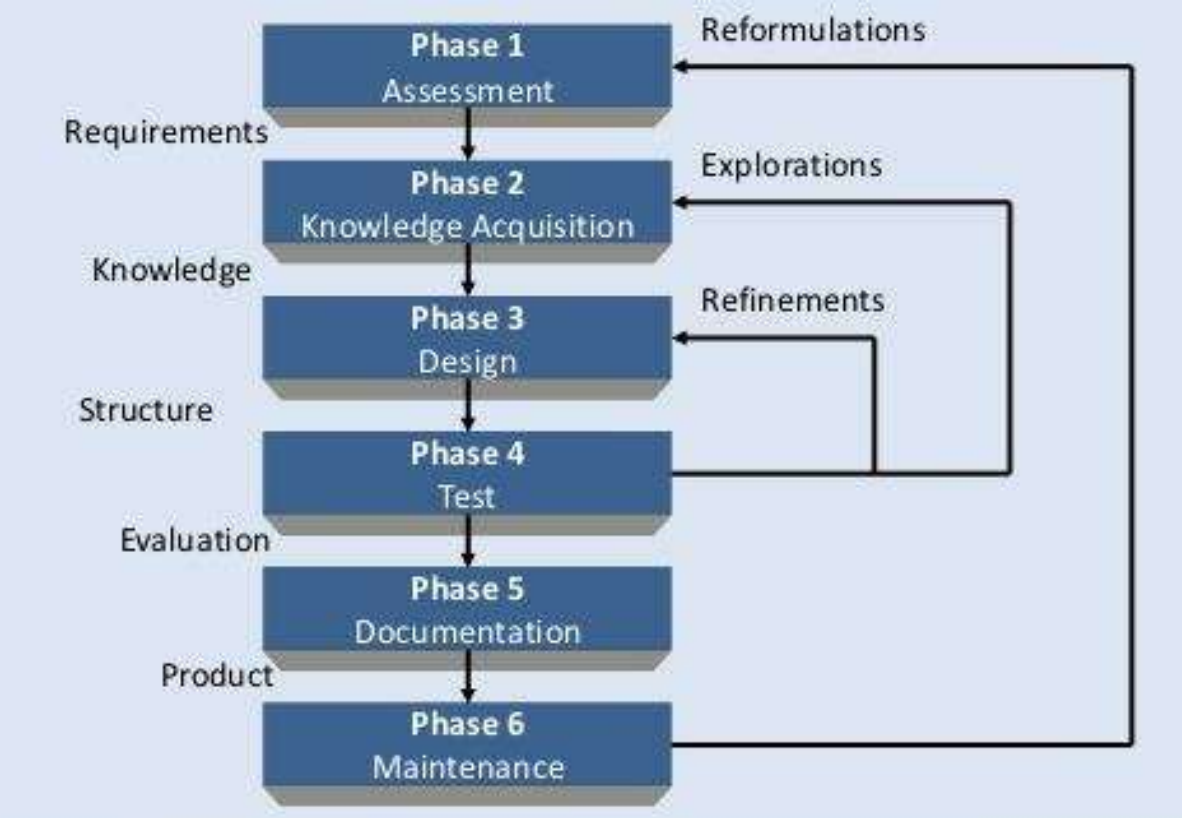
2.Uncertainty levels

3.Knowledge volume

4.User expertise

5.Explanation demands

6.Maintenance requirements



Expert System development life cycle

Quiz

1. Draw and label the basic architecture of an expert system.

2. Explain the function of the inference engine.

3. Differentiate knowledge base and working memory.

4. What is an expert system shell and why is it useful?

5. Discuss the role of explanation facilities in expert systems.

Knowledge Engineering (KE)

Knowledge Engineering (KE) is the discipline concerned with:

- •acquiring knowledge from domain experts,
- •representing and storing that knowledge,
- •verifying and validating it,
- •and integrating it into a usable Expert System.

✓ KE bridges the gap between **human expertise** and **computational representation**.

The Knowledge Engineer acts as intermediary between:

- **Domain Experts** (e.g., doctors, agronomists, loan officers)
- **End Users** (e.g., nurses, farmers, bank clerks)
- **Developers/Systems Engineers**

Domain Expert Interaction Challenges. Domain experts may:

- •omit key knowledge (tacit),
- •oversimplify procedures,
- •disagree with each other,
- •lack time for interviews.

Responsibilities of Knowledge Engineer:

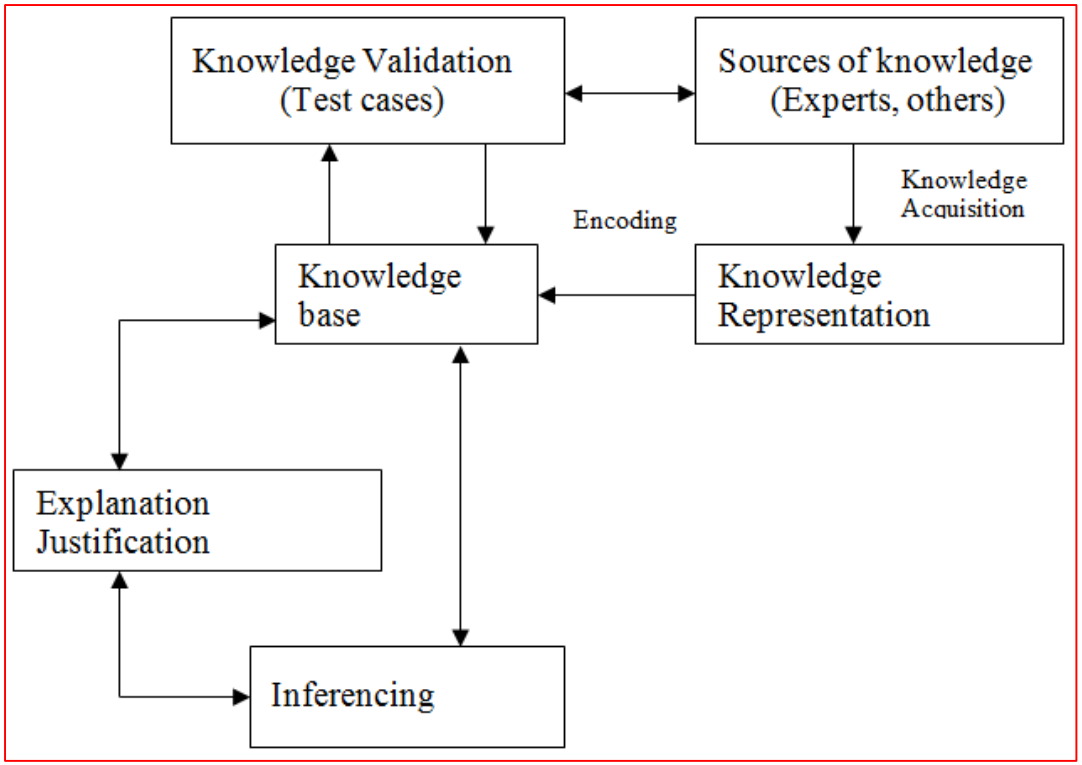
- identifying the problem domain,
- gathering expert knowledge,
- modeling and organizing that knowledge,
- selecting representation schemes,
- validating the knowledge,
- maintaining the system over time.

Knowledge Engineers must therefore be:

- patient,
- structured,
- diplomatically assertive,
- technically literate in both domains.

Knowledge Engineering Process/Lifecyle

1. Problem Identification
2. Domain Understanding
3. Knowledge Acquisition
4. Knowledge Representation
5. Prototype Development
6. Validation & Verification
7. Refinement & Maintenance
8. Deployment



✓ **This lifecycle often repeats iteratively.**

Problem Identification

Determine whether the problem is suitable for an Expert System. Suitable domains exhibit:

- ✓ heuristic rules,
- ✓ limited ambiguity,
- ✓ clear decision boundaries,
- ✓ repetitive reasoning tasks,
- ✓ expert scarcity.

Examples in Kenyan context:

- ✓ triage in county hospitals,
- ✓ livestock disease advisory,
- ✓ SACCO loan screening,
- ✓ agricultural pest diagnosis.

Unsuitable domains include:

- × open-ended research,
- × creative tasks,
- × high-dimensional prediction (better handled by ML).

Knowledge Acquisition

Sources of Knowledge:

1. Domain Experts (primary)
2. Documentation
3. Observation
4. Machine Learning & Data Mining
5. Case Studies
6. Decision Logs
7. Regulations/Policies (common in finance & law)
8. Interviews & Questionnaires

Acquisition Techniques:

1. structured interviews
2. think-aloud protocols
3. critical incident analysis
4. scenario-based questioning
5. card sorting
6. Delphi method
7. machine-assisted extraction

- ✓ Knowledge acquisition is often the hardest part due to the knowledge acquisition bottleneck.
  - Explicit Knowledge — documented and easily communicated
  - Tacit Knowledge — intuitive, experience-based, difficult to verbalize

Knowledge Verification & Validation

Checks internal correctness:
- rule consistency,
- rule completeness,
- conflict resolution,
- inference behavior.

Verification
• Conform to defined requirements and specificaticas
• Meet applicable standards
• Successfully complete each lifecycle activity before moving to thenxt
In short:
Verification ensures the system is built right.
Validation
• Meet the original requirements
• Solve the intended pro-blem
• Satisfy user needs in the real-world operational environment
In short:
Validation ensures the right system is built

Compares against real-world expert decisions:
- expert review,
- performance testing,
- field trials.

Maintenance & Refinement

- domains change (e.g., new crop pests),
- policies change (e.g., banking regulations),
- medical protocols update,
- user needs shift.

Maintenance tasks include:

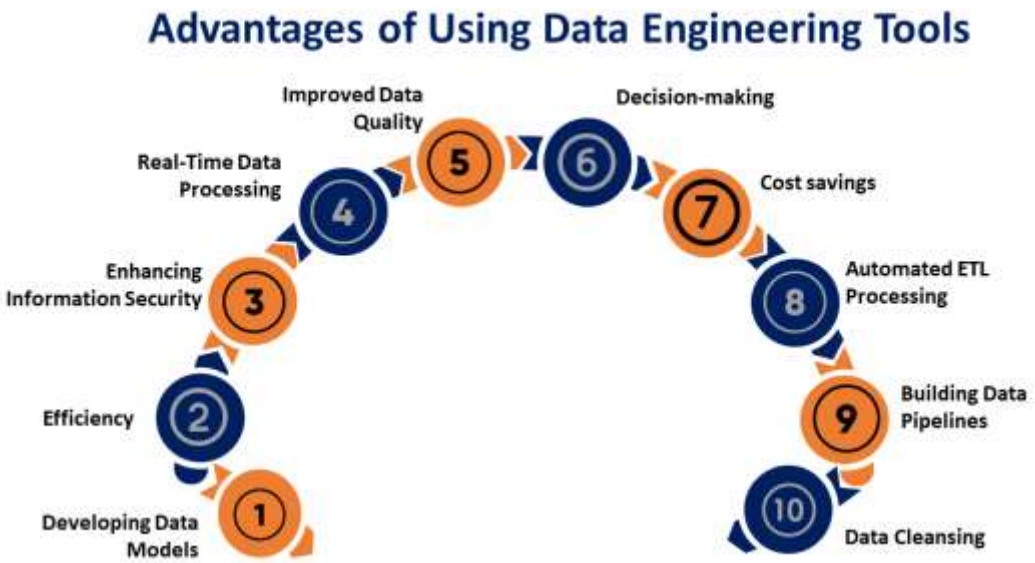
✓ adding/removing rules,
✓ updating heuristics,
✓ adjusting uncertainty values,
✓ expanding domain coverage,
✓ reorganizing knowledge base.

Poor maintenance leads to:

- outdated recommendations,
- loss of trust,
- system abandonment.

Tools for Knowledge Engineering

1. Expert System Shells
2. Rule Editors
3. Ontology Editors
4. Decision Table
5. Tools Modeling Languages (e.g., UML)
6. Explanation Debuggers



KE in Modern Systems

Modern KE integrates:
- knowledge graphs,
- ML models for knowledge extraction,
- hybrid reasoning engines,
- web APIs for distributed knowledge.

This is relevant to Kenyan sectors adopting AI in:
- ✓ fintech regulations,
- ✓ digital agriculture,
- ✓ public health triage,
- ✓ logistics & supply chain.

Quiz

1. What is knowledge engineering and why is it important?
2. Describe the knowledge acquisition bottleneck.
3. Differentiate verification and validation in Expert Systems.
4. Explain two techniques for acquiring tacit knowledge.
5. Describe the major phases of a knowledge engineering lifecycle.

UNCERTAINTY IN EXPERT SYSTEMS

- Real-world problem domains often involve **incomplete**, **imprecise**, or **uncertain** information. Human experts deal with uncertainty routinely using:
  - experience,
  - probabilities,
  - heuristics,
  - confidence levels,
  - linguistic terms (e.g., *"likely," "maybe," "mild," "severe"*).
- For Expert Systems to mimic human expertise, they must incorporate mechanisms to represent and reason under uncertainty.

Sources of Uncertainty

1. Incomplete Information - missing facts or symptoms e.g., patient cannot describe all symptoms
2. Noisy or Inaccurate Information - sensor inaccuracies, user errors
3. Ambiguous Knowledge - multiple explanations for same observation
4. Stochastic Processes - inherently probabilistic phenomena e.g., disease spread in livestock
5. Expert Disagreement - experts may classify differently
6. Non-determinism -future outcomes unknown

Need for Uncertainty Handling

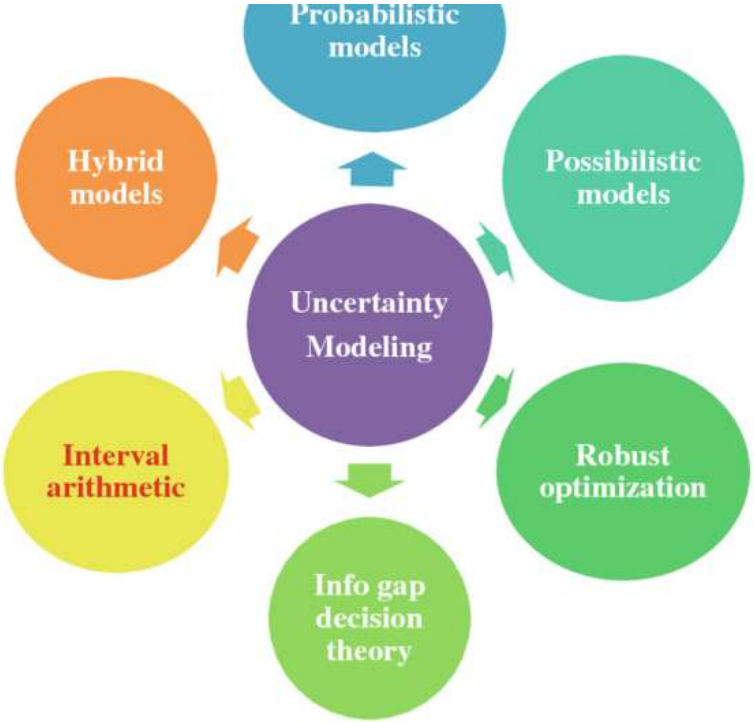
Expert Systems must support uncertainty to:

- model real expert reasoning
- provide realistic recommendations
- avoid brittle or overly deterministic decisions

Domains where uncertainty is critical include:

- ✓ medical diagnosis
- ✓ agriculture & veterinary systems
- ✓ financial risk analysis
- ✓ weather and environmental modeling
- ✓ fault diagnosis

Approaches/Frameworks to Handling Uncertainty



- Certainty Factors (CF)
- Bayesian Probability
- Fuzzy Logic
- Dempster–Shafer Theory
- Hybrid Systems

Certainty Factors (CF Model)

- Developed in the MYCIN medical expert system.

Representation

Each rule is assigned a CF between **–1 and +1**

- **+1** → definite belief
- **0** → no belief
- **–1** → definite disbelief

Example Rule

IF symptom = persistent_cough THEN disease = TB (CF = 0.6)

Meaning:

persistent_cough increases belief in TB by 60%

Combining CFs

If multiple symptoms support a conclusion, CFs are combined algorithmically.

Useful for medical systems in Kenyan context where:

- multiple symptoms overlap,
- lab testing may be limited.

Bayesian Probability

Bayesian reasoning is based on classical probability theory:

$$P(H \mid E) = \frac{P(E \mid H) \cdot P(H)}{P(E)}$$

Where:

- H = hypothesis (e.g., malaria)
- E = evidence (e.g., fever + chills)

Bayesian methods are useful when:

- statistical data is available,
- uncertainty is quantifiable,
- risk analysis is required.

Applications:

- ✓ credit scoring
- ✓ fraud detection
- ✓ clinical triage
- ✓ weather and climate forecasts

Fuzzy Logic

Fuzzy logic is used for vague or linguistic uncertainty, not probabilistic uncertainty.
Instead of binary true/false, fuzzy sets allow **degrees of membership**, e.g.:

- "temperature is high"
- "soil moisture is low"
- "risk is moderate"

Function Example

For soil moisture:

```
low:      0.8
medium:   0.3
high:     0.0
```

Why fuzzy logic is useful in Kenya:

- agricultural advice rarely precise ("water moderately")
- environmental data varies
- farmers use linguistic reasoning

Dempster–Shafer Theory (DST)

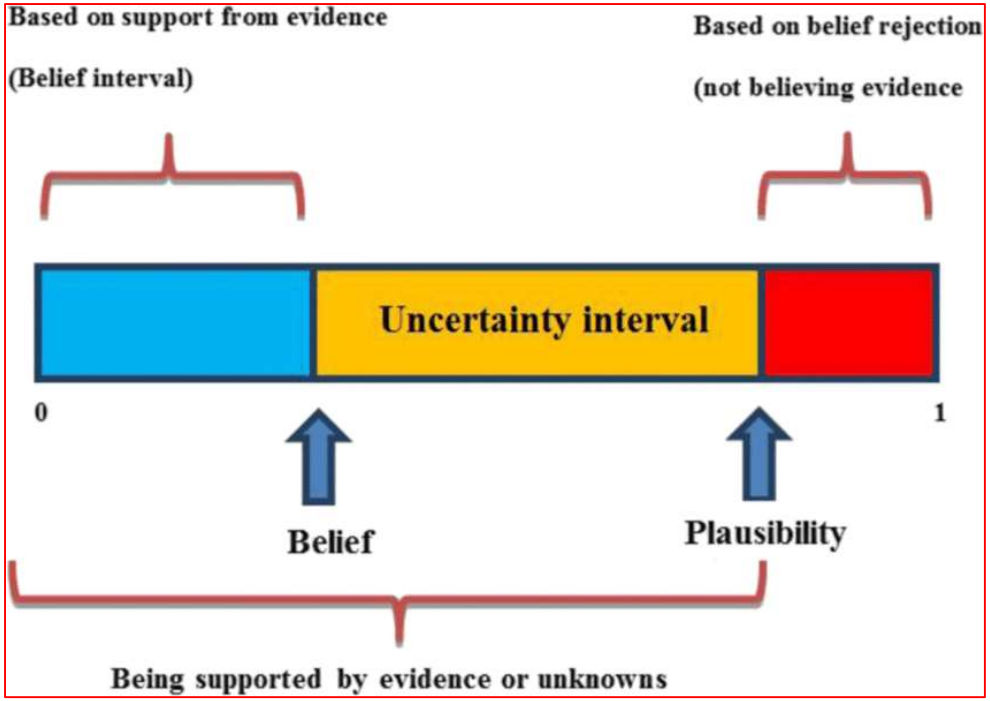
DST allows representation of:

- •belief,
- •disbelief,
- •ignorance.

Unlike Bayesian theory, DST does not require known prior probabilities.

Useful when:

- •data incomplete,
- •multiple sensors disagree,
- •expert disagreement high.



Comparative Summary

Approach	Handles	Needs Data?	Typical Domain
Certainty Factor	heuristic confidence	low	medicine
Bayesian	probabilistic uncertainty	high	finance
Fuzzy Logic	vagueness & linguistics	medium	agriculture
DST	belief & ignorance	medium	sensor fusion

Hybrid Uncertainty Systems

Modern Expert Systems integrate multiple models:

- •fuzzy + neural (neuro-fuzzy)
- •Bayesian + rules
- •classifier + rule engine

Hybridization improves:

- ✓ flexibility,
- ✓ robustness,
- ✓ accuracy.

Expert shells may support:

- •CF encoding (MYCIN style),
- •fuzzy rule bases,
- •Bayesian scoring,
- •or allow plugins for advanced reasoning.

Quiz

1. Define uncertainty and explain why it must be handled in Expert Systems.
2. Describe the Certainty Factor model and provide an example.
3. Differentiate probabilistic uncertainty from fuzzy uncertainty.
4. Explain Bayes' theorem in the context of diagnostic reasoning.
5. Discuss two real-world domains where uncertainty handling is essential.

Development Approach Overview

- Building an Expert System involves transforming domain expertise into a computational model capable of reasoning, advising, or diagnosing.
- Unlike conventional software, Expert System development centers on **knowledge acquisition** and **reasoning behavior**, rather than procedural programming alone.
- In academic and industrial settings, this process often leverages **Expert System Shells**, which provide ready-made inference engines and interfaces.

- **Building an Expert System typically involves**:STEPS
  1. Problem Selection
  2. Feasibility Analysis
  3. Domain Analysis
  4. Knowledge Acquisition
  5. Knowledge Representation
  6. Shell/Tool Selection
  7. Prototype Development
  8. Verification & Validation
  9. Refinement & Maintenance
  10. Deployment & User Training

Problem Selection

Not all problems are suitable for Expert Systems. Ideal characteristics:

- high expertise required
- domain relatively stable
- rules or heuristics exist
- symbolic reasoning possible
- expert scarcity exists
- clear decision criteria

Examples (Kenyan context):

- ✓ SACCO loan approval
- ✓ Maize disease diagnosis
- ✓ Veterinary triage
- ✓ Medical diagnostic support

Feasibility Analysis

Key questions:

1. Can experts articulate their heuristics?
2. Is uncertainty manageable?
3. Are there conflicting experts?
4. Will users accept automated advice?
5. Is expert knowledge available?
6. Is explainability required? (often yes in medicine/finance)

✓ Feasibility analysis prevents wasted development cycles.

Domain Analysis

Domain analysis identifies:

- vocabulary
- concepts
- relationships
- rules
- constraints
- goals

Tools include:

- conceptual maps
- semantic networks
- UML diagrams
- domain glossaries

Knowledge Acquisition

main outputs at this stage:

- ✓ rules
- ✓ cases
- ✓ decision tables
- ✓ ontologies
- ✓ fuzzy sets (if required)

Knowledge Representation

Domain	Likely Representation
Medical diagnosis	rules + CF
Agriculture	rules + fuzzy
Finance	Bayesian + rules
Legal/audit	logic + rules
Engineering	frames + rules

- Representation depends on domain

Shell / Tool Selection

Expert System Shells provide:

- •inference engine
- •rule editor
- •explanation module
- •debugging tools
- •UI widgets

Shell selection criteria:

1. Inference type (forward/backward)
2. Uncertainty support
3. Domain compatibility
4. User interface flexibility
5. Deployment environment
6. Cost/licensing

Examples of Shells

- •**CLIPS** (NASA origin, open source)
- •**JESS** (Java-based)
- •**KBMS** platforms
- •**Prolog-based shells**
- •**Custom in-house shells**
- •**Medical shells** (commercial)

Prototype Development

Typical prototype workflow:

1. Load knowledge base
2. Initialize working memory
3. Run inference cycle
4. Display conclusions
5. Show explanation trace

Prototype goals:

- validate domain knowledge
- identify missing rules
- test usability
- demonstrate feasibility

Verification & Validation

Verification checks internal correctness:

- rule completeness
- rule consistency
- conflict handling
- inference behavior

Validation tests external correctness:

- expert review
- case testing
- field trials
- user feedback

✓ Validation is critical in domains with safety implications (medicine, veterinary, finance).

Refinement & Maintenance

Refinement addresses:

- ✓ missing rules
- ✓ inaccurate heuristics
- ✓ evolving domain knowledge
- ✓ new policy constraints (common in finance & law)
- ✓ new diseases/pests (common in agriculture)

✓ Maintenance is ongoing — Expert Systems age if not updated.

Deployment & User Training

Deployment requires:

- documentation,
- training,
- user manuals,
- interface adjustments.

✓ User acceptance is often the hardest part of deployment — especially in expert-dominated domains where professionals may feel challenged by automation.

Evaluation Criteria/Metrics for Expert Systems

1. **Correctness** — matches expert reasoning
2. **Completeness** — sufficient domain coverage
3. **Consistency** — no contradictions
4. **Efficiency** — acceptable performance
5. **Explainability** — users understand outputs
6. **Usability** — interface accessible
7. **Robustness** — handles missing or noisy input
8. **Maintainability** — easy to update

Case Example - SACCO Loan Approval

Rules may include:

> IF income ≥ threshold AND payslip=verified AND collateral=adequate THEN approveLoan (CF=0.7)

> IF CRB=blacklisted THEN rejectLoan (CF=1.0)

> IF income borderline AND collateral weak THEN requestGuarantor

Notice uncertainty and exceptions embedded — reflecting real banking practice.

✓ Notice uncertainty and exceptions embedded — reflecting real banking practice.

Quiz

1. Describe five steps involved in building an Expert System prototype.
2. Explain the role of an Expert System Shell.
3. Distinguish between feasibility analysis and validation.
4. Why is user acceptance important in deployment stages?
5. Discuss three characteristics of suitable problem domains for Expert Systems.

History – Evolution of ES

- Expert Systems originated in the 1960s and matured in the 1980s, but their evolution continues — especially as they integrate with modern AI techniques such as:
  - Machine Learning (ML),
  - Natural Language Processing (NLP),
  - Knowledge Graphs,
  - Cloud Computing,
  - and Explainable AI (XAI).
- Modern applications extend across diverse industries

- Classical Expert Systems relied primarily on:
  - ✓ symbolic rules
  - ✓ heuristics
  - ✓ human-crafted knowledge
- Modern developments introduce:
  1. **Hybrid Systems:** symbolic + statistical
  2. **Neuro-Symbolic Systems:** neural networks + logic
  3. **Explainable AI (XAI):** interpretable ML models
  4. **Knowledge-Based Decision Support Systems**
  5. **Knowledge Graph–Driven Systems**

Integration with Machine Learning

Unlike pure ML models, Expert Systems provide:

- ✓ interpretability
- ✓ explainability
- ✓ structured reasoning

ML integration introduces:

- •pattern recognition,
- •anomaly detection,
- •data-driven knowledge extraction.

Example hybrid use case:

ML classifies plant disease probability → ES recommends treatment actions.

Industry Applications

(A) Healthcare & Medical Diagnosis

Applications include:

•symptom triage,
•diagnostic suggestions,
•treatment recommendation,
•drug interaction checking.

In Africa, used for:

✓ community health triage
✓ infectious disease decision support
✓ maternal health advisory tools

(B) Agriculture & Veterinary Systems

Expert Systems support:

•crop disease identification,
•livestock disease advisory,
•fertilizer recommendations,
•irrigation scheduling (fuzzy logic),
•post-harvest quality grading.

Kenyan context:

✓ maize & horticulture advisory tools
✓ dairy veterinary triage
✓ coffee/tea plant advisory systems

(C) Financial Services & Fintech

Expert Systems assist in:

•loan approval,
•credit scoring,
•fraud detection,
•compliance auditing,
•risk assessment.

Kenya's strong mobile fintech ecosystem (MPESA, SACCOs, microfinance) makes this highly relevant.

(D) Legal, Audit & Compliance Automation

Rule-based knowledge systems suit:

•regulation checking,
•policy compliance,
•contract review,
•tax advisory.

Common in enterprise and public sector environments.

Industry Applications

(E) Industrial & Manufacturing
Applications include:
•machine fault diagnosis,
•maintenance scheduling,
•production control,
•quality inspection.
Fuzzy and hybrid systems used for sensor-based reasoning.

(F) Environmental & Climate Modelling
Systems assist in:
•drought impact forecasting,
•pest outbreak prediction,
•wildfire risk scoring.

Application	Description
Design Domain	Camera lens design, automobile design.
Medical Domain	Diagnosis Systems to deduce cause of disease from observed data, conduction medical operations on humans.
Monitoring Systems	Comparing data continuously with observed system or with prescribed behavior such as leakage monitoring in long petroleum pipeline.
Process Control Systems	Controlling a physical process based on monitoring.
Knowledge Domain	Finding out faults in vehicles, computers.
Finance/Commerce	Detection of possible fraud, suspicious transactions, stock market trading, Airline scheduling, cargo scheduling.

Recent Research Directions

- **Explainable AI (XAI) -** bridging ML opacity with ES interpretability
- **Knowledge Graphs & Ontologies -** semantic reasoning for large data ecosystems
- **Neuro-Symbolic Systems -**combining neural networks with logic reasoning
- **Edge Deployment -**running expert logic on mobile devices for field use. Useful in rural agriculture & health scenarios
- **Cloud-Based Knowledge Platforms -** shared domain knowledge bases across stakeholders
- **Crowdsourced Knowledge Acquisition -**distributed gathering of expert rules (e.g., farmer networks)

Challenges/Obstacles in Modern Deployment

- knowledge acquisition bottleneck,
- model updating and maintenance,
- expert disagreement,
- trust and explainability barriers,
- integration with legacy systems,
- data quality inconsistencies,
- regulatory and ethical concerns.

Trust and explainability remain critical in domains such as:

- ✓ medicine
- ✓ finance
- ✓ law

Advantages of Modern Adoption

✓ 24/7 availability

✓ low marginal cost

✓ consistent decisions

✓ improved reach of expertise

✓ training benefits (educational explainability)

✓ structured documentation of domain knowledge

Quiz

1. Describe two modern trends in Expert System development.
2. Discuss financial industry applications of Expert Systems.
3. Explain why explainability remains important in modern AI systems.
4. Describe three challenges facing Expert System deployment in developing countries.

Lab Objectives

By the end of this lab, students should be able to:

- Understand the structure of a Knowledge-Based System
- Navigate an Expert System Shell interface
- Load and run an existing expert system
- Observe inference processes and explanations

Tools / Software (Indicative)

- Expert System Shell (e.g. CLIPS / JESS / Prolog-based shell)
- PC with Java / interpreter (depending on shell)
- Sample KBS provided by instructor

Lab Section Summary

✓ Hands-on experience with KBS
✓ Practical rule development
✓ Inference observation
✓ Interface design
✓ End-to-end Expert System project

Lab 1 : Running a Knowledge-Based System (KBS) & Learning the Interface

Task 1: Launch the Expert System Shell

1.Start the shell environment

2.Identify main interface components:

1. rule editor
2. working memory viewer
3. inference control
4. explanation window

Task 2: Load an Existing Knowledge Base

1.Load a sample expert system

2.Identify:

1. facts
2. rules
3. goals (if any)

Task 3: Run the Expert System

1.Provide required inputs

2.Observe:

1. rule firing sequence
2. facts added to working memory
3. final conclusion

Task 4: Examine Explanations

1.Use the explanation facility

2.Answer:

1. Why was a particular question asked?
2. How was the conclusion reached?

Lab 1 Deliverables

•Screenshot(s) of system output

•Short report:

- observations on inference process
- description of UI components

Lab 2: Knowledge Acquisition, Inference & Decision Structures

Students will:
- Acquire simple domain knowledge
- Encode rules in an Expert System
- Apply forward & backward chaining
- Use dependency diagrams and decision tables

Lab 3 Deliverables
- Interface screenshots
- Description of user interaction flow
- Explanation facility demonstration

Part A: Knowledge Acquisition

Task 1

Select a simple domain (examples):
- crop disease diagnosis
- loan approval rules
- equipment fault diagnosis

Task 2

Interview or simulate an expert and extract:
- conditions
- conclusions
- exceptions

Part B: Rule Encoding

Create at least **5 IF–THEN rules**.

Example:

IF fever AND chills THEN suspect malaria.

Part C: Forward & Backward Chaining

Task 1: Forward Chaining
- Run the system using input facts
- Observe rule firing order

Task 2: Backward Chaining
- Define a goal
- Observe sub-goal generation

Part D: Dependency Diagrams

Draw diagrams showing:
- relationships between facts
- rule dependencies

Part E: Decision Tables

Convert rule sets into decision tables.

Lab 3: User Interface Development

Students will:
•Design a user-friendly interface
•Create input forms
•Display results and explanations

Task 1: Interface Design
•Identify user input fields
•Define prompts/questions
•Design output messages

Task 2: Implement Interface
•Use shell UI tools or external interface (if supported)
•Connect UI inputs to working memory

Task 3: Explanation Display
•Enable “Why” and “How” explanations
•Test with different inputs

Lab 3 Deliverables
•Interface screenshots
•Description of user interaction flow
•Explanation facility demonstration

Project Development & Presentations

Lab 4 Objectives
Students will:
•Build a complete Expert System prototype
•Demonstrate reasoning and explanations
•Present design decisions

Presentation Guidelines
•10–15 minutes per group
•Live demonstration
•Questions from panel
•Peer feedback

Project Requirements

Each project must include:

✓ problem definition

✓ domain description

✓ rule base (minimum 10 rules)

✓ inference strategy used

✓ uncertainty handling (optional but rewarded)

✓ user interface

✓ explanation facility

Assessment Rubric

Component	Marks
Problem suitability	10
Knowledge representation	20
Inference strategy	20
System correctness	20
Interface & explanation	15
Presentation	15